## OCS001- COMPUTER SCIENCE I

| |
|---|
| **Credit Hours:** 3-4 Semester Hours |
| **Pre-Requisite:** Fundamentals of Computing or Proficiency |
| **Related TAG:** Computer Science |
| **Student Learning Outcomes marked with an asterisk (*) are considered essential and must be covered:** |
| **Learning outcomes 1.** Ability to apply computing theories and use basic software applications to solve problems. |
| **Learning outcomes 2.** Design solutions focused algorithms using pseudocode and/or flowchart to solve programming problems. * |
| **Learning outcome 3.** Identify the data types available and apply this knowledge to declare and use variables and constants. * |
| **Learning outcome 4.** Demonstrate the ability to use operators to create logical expressions, mathematical calculations, and assignment statements. * |
| **Learning outcome 5.** Build conditional logic using Boolean expressions and decision structures. * |
| **Learning outcomes 6.** Construct loops to implement iterative logic. * |
| **Learning outcomes 7.** Ability to create and effectively use subroutines (methods/functions/procedures). * |
| **Learning outcomes 8.** Use a data structure, such as an array, to store and manipulate a collection of related elements. * |
| **Learning outcomes 9:** Properly use error checking or proper testing of program to debug, validate data, and resolve errors. * |

## OCS002- COMPUTER SCIENCE II

| |
|---|
| **Credit Hours:** 3-4 Semester Hours |
| **Related TAG:** Computer Science |
| **Student Learning Outcomes marked with an asterisk (*) are considered essential and must be covered:** |
| **Learning outcome 1.** Explain the core principles of object-oriented programming. * |
| **Learning outcome 2.** Demonstrate the ability to create user-defined types (classes) and instantiate programming objects. * |
| **Learning outcome 3.** Create object-oriented solutions that use class inheritance to create subclasses. * |
| **Learning outcomes 4.** Utilize advanced object-oriented techniques, such as abstract classes and interfaces. * |
| **Learning outcomes 5.** Design solutions implementing simple data structures such as a linked list or queue, to store and manage a collection of values. * |
| **Learning outcomes 6.** Implement simple search and sort algorithms and explain the differences in their time complexities. * |
| **Learning outcome 7.** Demonstrate the ability to evaluate algorithms, to select from a range of possible options and to implement the algorithm in a particular context. * |
| **Learning outcomes 8.** Analyze and apply recursive programming techniques. * |